

REMARKS

In the Office Action, the Examiner indicated that claims 1-17 are pending in the application and the Examiner rejected all claims. The rejections are respectfully traversed below.

I. Rejection of Claims 1- 17 under 35 U.S.C. §103(a)

At pages 2-7 of the Office Action, the Examiner rejects claims 1-17 under 35 U.S.C. §103(a) based on a combination of Rose et al. (U.S. Patent No. 5,752,244) and Bowman-Anuah (U.S. Patent No. 6,256,773).

A. The Present Invention

The present invention is an asset locator (search engine) for locating software assets, code assets and the like that are stored in code repositories used by software designers. It provides the capability for the gathering of information about assets contained in the code repositories and the capturing of the gathered information in a database that can be used for the conducting of subsequent searches. The present invention has particular application in a software-development environment where the stored code assets may number in the millions and may be written in diverse languages such as, for example, Java, C/C + +, COBOL, HTML, and/or XML.

A crawl process is performed on a storage device on which assets are stored to identify the assets. Asset-specific parameters related to the stored assets are identified, and the assets are then analyzed based upon these parameters. Textual and semantic

information is extracted from the stored assets and then the extracted textual and semantic information is stored and indexed for retrieval.

In a preferred embodiment, a series of data analyzers that are specific to each type of data contained in the code repositories (e.g., a Java analyzer, a C/C++ analyzer, a COBOL analyzer, an HTML analyzer, and/or an XML analyzer) are integrated into the system so that they can be used to search the code repositories using particular attributes specific to the semantics of a particular language used to code the asset. In another preferred embodiment, the repositories are crawled automatically according to a schedule defined by the user, and the results of the crawling are stored in a database. Ordinary keyword searching can then be used with the system, either independently or combined with the attribute-specific semantic searching, to search the database.

B. Rose et al.

U.S. Patent No. 5,752,244 to Rose et al. ("Rose") teaches a method, apparatus, and article of manufacture providing computerized management of multimedia assets of various types, including image, video, audio, text, and program code media types. Multimedia assets are checked into a computer system along with specified characteristics and identification information for the assets. Each asset is categorized according to a plurality of asset characteristics and asset identification information using predefined fields for the characteristics and identification information. In this manner, common characteristic and identification information is available for all files regardless of their type. The information to be searched is input manually and the resulting database is searched using web browser 46.

C. Bowman-Anuah

U.S. Patent No. 6,256,773 to Bowman-Anuah teaches a system, method and article of manufacture for affording consistency in a development architecture framework as components in the framework change. The Examiner relies upon Bowman-Anuah solely for the disclosure in column 4, lines 22-29, repeated herein in its entirety:

“OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.”

D. The Examiner has not Established a *prima facie* Case of Obviousness

To support a rejection under 35 U.S.C. §103, a reason, suggestion, or motivation to lead an inventor to combine two or more references must be found. *Pro-Mold and Tool Co. v. Great Lakes Plastics Inc.*, 37 U.S.P.Q.2d 1627, 1629 (Fed.Cir. 1996). The Examiner has not met her burden in establishing a reason, suggestion, or motivation for combining the cited references.

The Examiner asserts that claims 1-17 are unpatentable under 35 U.S.C. § 103(a) based on Rose et al. in view of Bowman-Anuah.

While the Examiner has made assertions regarding the claims and alleges teaching of various claim elements in Rose and/or Bowman-Anuah, a reading of the cited portions of the patents does not support the Examiner's conclusion.

Applicant stands by and reiterates the arguments presented in the reply filed on February 4, 2003 in response to the Office Action of October 4, 2002. In particular, applicant focuses the Examiner's attention again on her assertion that Figure 4 of Rose identifies the performance of a crawl process.

The Examiner states in the Office Action mailed March 20, 2003 that "Rose clearly discloses performance of a crawl process as a web browser 46 at fig. 4, and the definition of a crawl is a web browser in Microsoft Computer Dictionary, fifth Edition." The Examiner misreads the Microsoft Dictionary and is simply incorrect. First, a copy of the page from the Microsoft Dictionary relied upon by the Examiner is attached hereto. The following portions from page 133 of the Microsoft Computer Dictionary, fifth Edition, are relevant:

crawl *vb.* To compile and organize entries for a search engine by reading Web pages and related information. Crawling is typically performed by programs called "spiders."

crawler *n.* See spider, Web browser.

From this definition, the Examiner concludes that a web browser is a crawler. However, this is an incorrect reading of the dictionary page. By using the term "*See*" the reader is simply directed to the terms "spider" and "Web browser" for explanation. The Examiner did not include or give the definition of "spider" or "Web browser" from the Microsoft Computer Dictionary. However, as is clear from the definition of the term "crawl" the definition given in the Microsoft Dictionary is consistent with the argument provided in the response filed by the application on February 4, 2003. Specifically, as is asserted in the previous response, a web crawler is used to build a database. Applicant

provides, for further explanation, pages 682 and 720 of "Computer Networks", Third edition, by Andrew S. Tanenbaum (copyright 1996 by Prentice Hall PTR) in which the distinctions between the crawling and searching capability of a web crawler or spider is contrasted to the ability of a browser simply to fetch a requested page, i.e., it will search and find a specific URL input thereto, but will not crawl a database or series of databases looking for and/or building a database consisting of the search terms. This is a clear distinction; Rose does disclose the use of a web browser, but does not disclose or suggest the use of a web crawler.

Applicant reiterates all of the arguments presented in its February 4, 2003 response; a Notice of Appeal is filed concurrently herewith to appeal the issues, in the event that the Examiner continues to maintain the present rejections.

II. The Drawing Correction

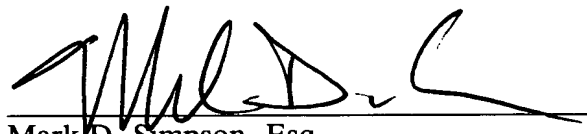
As required on page 1 of the Office Action (Office Action Summary), applicant encloses herewith a corrected Figure 2, which drawing correction was submitted to the Examiner for approval with applicant's Reply, dated August 12, 2002, to the Examiner's Action mailed May 10, 2003. This correction was approved by the Examiner in her Action mailed October 4, 2002.

III. Conclusion

The rejection of claims 1-17 have been traversed. Accordingly, reconsideration of the present application, and withdrawal of the rejections on the grounds of 35 U.S.C. §103 is respectfully requested.

Respectfully submitted,

JUNE 29 2003
Date


Mark D. Simpson, Esq.
Registration No. 32,942
SYNNESTVEDT & LECHNER LLP
Suite 2600 Aramark Tower
1101 Market Street
Philadelphia, PA 19107
Telephone: (215) 923-4466
Facsimile: (215) 923-2189

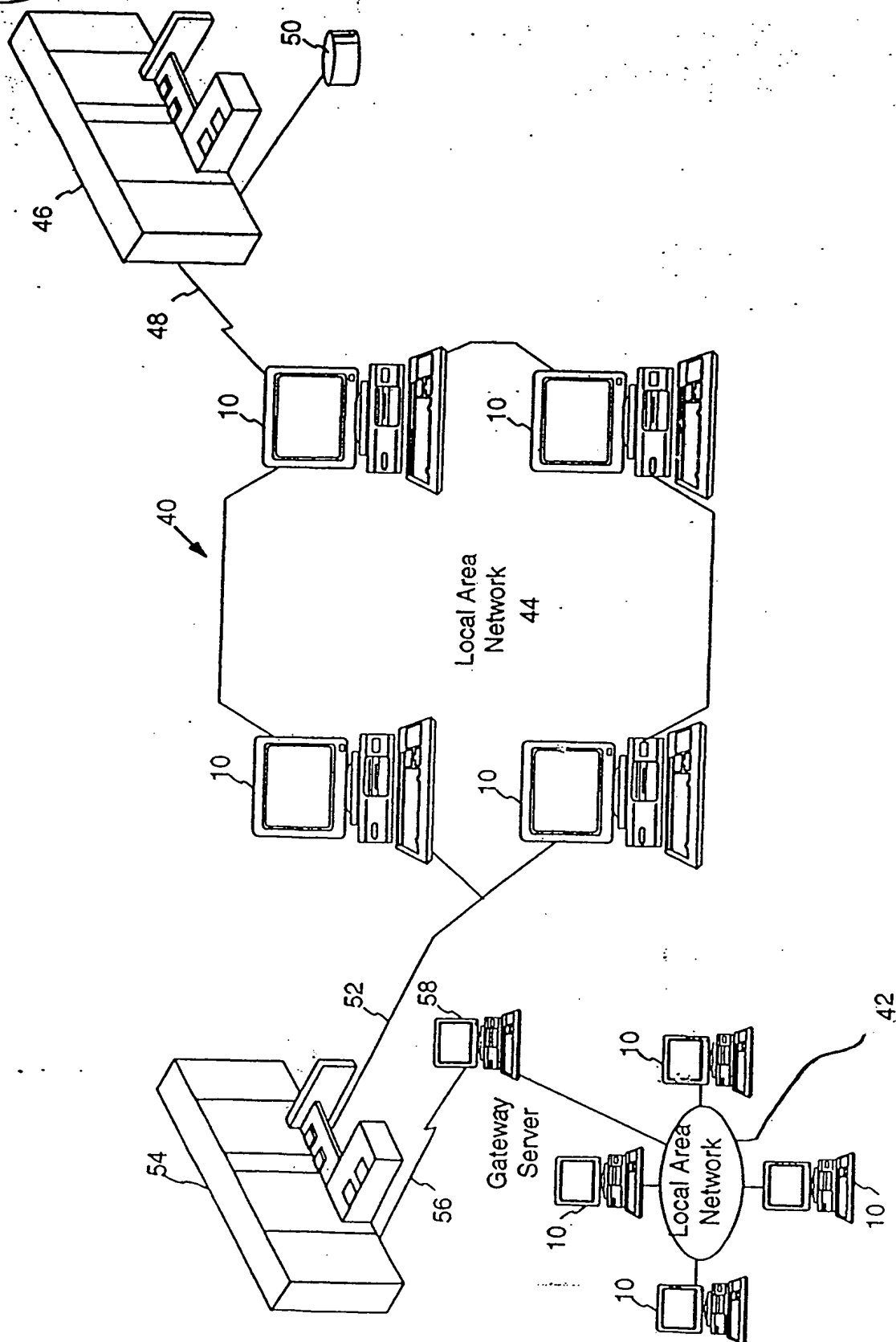


FIG. 2

From: Andy Nguyen
To: Mark Simpson

09/473 554

cradle

critical path method



usually from a computer system or use computer resources. However, the goal of the majority is only to break into the system. *See also* hacker (definition 2).

cradle *n.* A receptacle used to recharge the batteries in some handheld or palm-size PCs or PDAs (personal digital assistants). Some cradles also serve as a means to connect these smaller devices with a desktop PC. Not all of these devices require a cradle to recharge or connect to a desktop system. *Also called:* dock, docking station.

cramfs *n.* Short for Compressed Read-Only File System and **cram** a filesystem onto a small ROM. A filesystem feature available with Linux version 2.4 systems. Cramfs are used in handheld Linux devices to compress and write applications to ROM or Flash memory.

crash¹ *n.* The failure of either a program or a disk drive. A program crash results in the loss of all unsaved data and can leave the operating system unstable enough to require restarting the computer. A disk drive crash, sometimes called a disk crash, leaves the drive inoperable and can cause loss of data. *See also* abend, head crash.

crash² *vb.* 1. For a system or program, to fail to function correctly, resulting in the suspension of operation. *See also* abend. 2. For a magnetic head, to hit a recording medium, with possible damage to one or both.

crash recovery *n.* The ability of a computer to resume operation after a disastrous failure, such as the failure of a hard drive. Ideally, recovery can occur without any loss of data, although usually some, if not all, data is lost. *See also* crash¹.

crawl *vb.* To compile and organize entries for a search engine by reading Web pages and related information. Crawling is typically performed by programs called "spiders."

crawler *n.* *See* spider, Web browser.

Cray-1 *n.* An early supercomputer developed in 1976 by Seymour Cray. Extremely powerful in its day, the 64-bit Cray-1 ran at 75 MHz and was capable of executing 160 million floating-point operations per second. *See also* supercomputer.

CRC *n.* Acronym for cyclical (or cyclic) redundancy check. A procedure used in checking for errors in data transmission. CRC error checking uses a complex calculation to generate a number based on the data transmitted. The sending device performs the calculation before transmission and includes it in the packet that it sends to the

receiving device. The receiving device repeats the same calculation after transmission. If both devices obtain the same result, it is assumed that the transmission was error free. The procedure is known as a redundancy check because each transmission includes not only data but extra (redundant) error-checking values. Communications protocols such as XMODEM and Kermit use cyclical redundancy checking.

create method *n.* In Java programming, a method defined in the home interface and invoked by a client to create an enterprise java bean. *See also* enterprise java bean, method.

creator *n.* On the Apple Macintosh, the program that creates a file. Files are linked to their creators by creator codes; this link enables the operating system to open the creator application when a document file is opened.

credentials *n.* A set of information that includes identification and proof of identification that is used to gain access to local and network resources. Examples of credentials are user names and passwords, smart cards, and certificates.

creeping featurism *n.* The process by which features are added to a new version of a program by software developers until the program becomes unduly cumbersome and difficult to use. Generally, creeping featurism occurs as developers attempt to enhance the competitiveness of the program with each new release by adding new features.

crippled version *n.* A scaled-down or functionally reduced version of hardware or software, distributed for demonstration purposes. *See also* demo.

critical error *n.* An error that suspends processing until the condition can be corrected either by software or by user intervention (for example, an attempt to read to a nonexistent disk, an out-of-paper condition on the printer, or a checksum fault in a data message).

critical-error handler *n.* A software routine that attempts to correct or achieve a graceful exit from a critical or threatening error. *See also* critical error, graceful exit.

critical path method *n.* A means of evaluating and managing a large project by isolating tasks, milestone events, and schedules and by showing interrelationships among them. The critical path for which this method is named is a line connecting crucial events, any of which, if delayed, affects subsequent events and, ultimately, completion of the project. *See also* PERT.

RECEIVED

JUN 26 2003

133 Technology Center 2100

FOR FILE	23546125A
RECEIVED	
JUN 20 2003	
SYNNSTVEDT & LECHNER	
ATTN: MDS	

BEST AVAILABLE COPY

system, we will discuss both the client (i.e., user) side and the server side. Then we will examine the language in which Web pages are written (HTML and Java). Finally, comes an examination of how to find information on the Web.

7.6.1. The Client Side

From the users' point of view, the Web consists of a vast, worldwide collection of documents, usually just called **pages** for short. Each page may contain links (pointers) to other, related pages, anywhere in the world. Users can follow a link (e.g., by clicking on it), which then takes them to the page pointed to. This process can be repeated indefinitely, possibly traversing hundreds of linked pages while doing so. Pages that point to other pages are said to use **hypertext**.

Pages are viewed with a program called a **browser**, of which Mosaic and Netscape are two popular ones. The browser fetches the page requested, interprets the text and formatting commands that it contains, and displays the page, properly formatted, on the screen. An example is given in Fig. 7-58(a). Like many Web pages, this one starts with a title, contains some information, and ends with the email address of the page's maintainer. Strings of text that are links to other pages, called **hyperlinks**, are highlighted, either by underlining, displaying them in a special color, or both. To follow a link, the user places the cursor on the highlighted area (using the mouse or the arrow keys) and selects it (by clicking a mouse button or hitting ENTER). Although nongraphical browsers, such as Lynx, exist, they are not as popular as graphical browsers, so we will concentrate on the latter. Voice-based browsers are also being developed.

Users who are curious about the Department of Animal Psychology can learn more about it by clicking on its (underlined) name. The browser then fetches the page to which the name is linked and displays it, as shown in Fig. 7-58(b). The underlined items here can also be clicked on to fetch other pages, and so on. The new page can be on the same machine as the first one, or on a machine halfway around the globe. The user cannot tell. Page fetching is done by the browser, without any help from the user. If the user ever returns to the main page, the links that have already been followed may be shown with a dotted underline (and possibly a different color) to distinguish them from links that have not been followed. Note that clicking on the *Campus Information* line in the main page does nothing. It is not underlined, which means that it is just text and is not linked to another page.

Most browsers have numerous buttons and features to make it easier to navigate the Web. Many have a button for going back to the previous page, a button for going forward to the next page (only operative after the user has gone back from it), and a button for going straight to the user's own home page. Most browsers have a button or menu item to set a bookmark on a given page and another one to display the list of bookmarks, making it possible to revisit any of

BEST AVAILABLE COPY

bytes, individual frames can use a strong error-correcting code, and all frames can be sent two or three times. Many other covert channels exist, and it is extremely difficult to discover and block them all. For more information about the security problems in Java see (Dean and Wallach, 1995).

In short, Java introduces many new possibilities and opportunities into the World Wide Web. It allows Web pages to be interactive, and to contain animation and sound. It also permits browsers to be infinitely extensible. However, the Java model of downloading applets also introduces some serious new security problems that have not been entirely solved yet.

7.6.5. Locating Information on the Web

Although the Web contains a vast amount of information, finding the right item is not always easy. To make it easier for people to find pages that are useful to them, several researchers have written programs to index the Web in various ways. Some of these have become so popular that they have gone commercial. Programs that search the Web are sometimes called **search engines**, **spiders**, **crawlers**, **worms**, or **knowbots** (knowledge robots). In this section we will give a brief introduction to this subject. For more information, see (Pinkerton, 1994; and McBryan, 1994).

Although the Web is huge, reduced to its barest essentials, the Web is a big graph, with the pages being the nodes and the hyperlinks being the arcs. Algorithms for visiting all the nodes in a graph are well known. What makes Web indexing difficult is the enormous amount of data that must be managed and the fact that it is constantly changing.

Let us start our discussion with a simple goal: indexing all the keywords in Web pages' titles. For our algorithm, we will need three data structures. First, we need a large, linear array, *url_table*, that contains millions of entries, ultimately one per Web page. It should be kept in virtual memory, so parts not heavily used will automatically be paged to disk. Each entry contains two pointers, one to the page's URL and one to the page's title. Both of these items are variable length strings and can be kept on a heap (a large unstructured chunk of virtual memory to which strings can be appended). The heap is our second data structure.

The third data structure is a hash table of size n entries. It is used as follows. Any URL can be run through a hash function to produce a nonnegative integer less than n . All URLs that hash to the value k are hooked together on a linked list starting at entry k of the hash table. Whenever a URL is entered into *url_table*, it is also entered into the hash table. The main use of the hash table is to start with a URL and be able to quickly determine whether it is already present in *url_table*. These three data structures are illustrated in Fig. 7-75.

Building the index requires two phases: searching and indexing. Let us start with a simple engine for doing the searching. The heart of the search engine is a recursive procedure *process_url*, which takes a URL string as input. It operates as



Computer Networks

Third Edition

Andrew S. Tanenbaum

*Vrije Universiteit
Amsterdam, The Netherlands*

RECEIVED

JUN 26 2003

Technology Center 2100

For book and bookstore information



<http://www.prenhall.com>

BEST AVAILABLE COPY



*Prentice Hall PTR
Upper Saddle River, New Jersey 07458*